**Visualization Library Documentation**

**Objective:** To create a comprehensive documentation guide for 2 given python visualisation libraries (matplotlib & pandas).

1. **Matplotlib:**

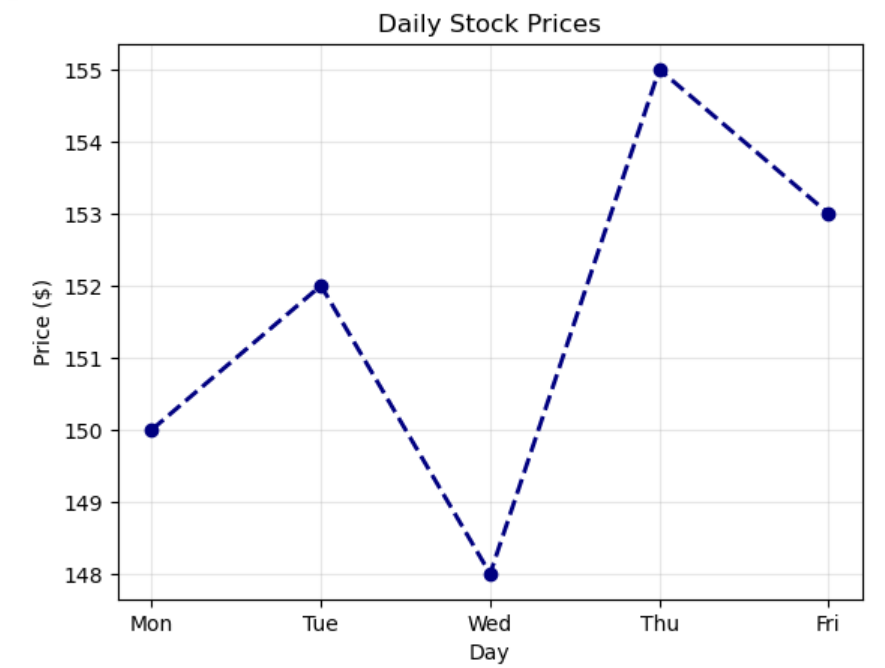
* Provides a low-level interface for creating static, animated, or interactive visualizations. It is the backbone of many higher-level libraries
* Unique Features-
* Granular Control (Adjust every plot element)
* Flexibility (Supports 2D/3D plots, animation, and custom extensions)
* Publication-Ready Output (Export plots in PDF, SVG, or PNG formats with high resolution.

**Graph Types:**

1. *Line Plot:*

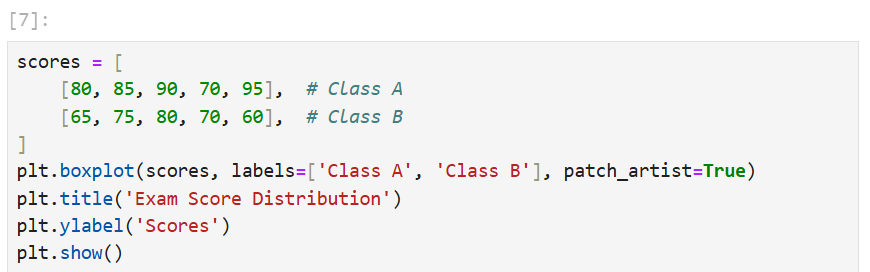
* A line plot connects data points with straight lines, ideal for continuous data trends. It emphasizes movement or progression.
* Use case: Tracking stock prices over a week.
* Code Snippet:

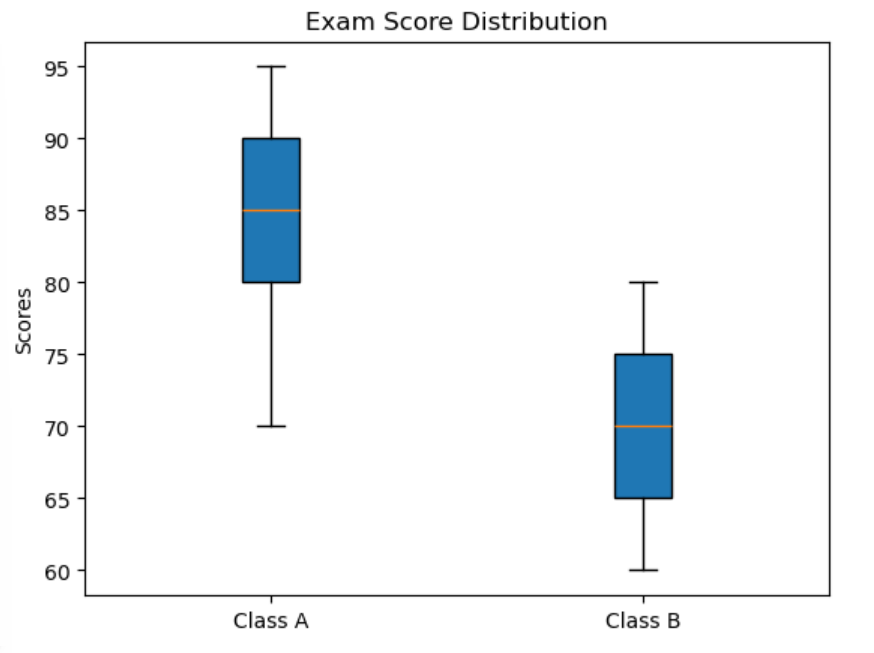


* Line plot output:
* Interpretation: The line’s slope indicates trends (e.g., a steep rise from Wed to Thu suggests a price surge).

1. *Box Plot:*

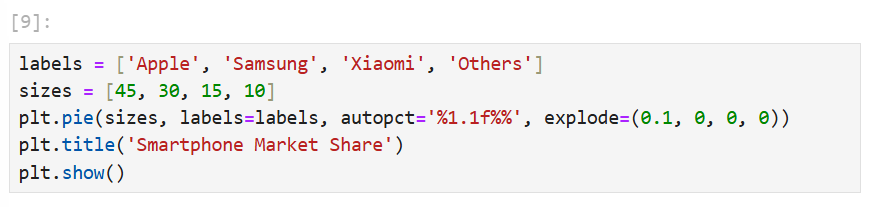
* Displays data distribution using quartiles. The box spans the interquartile range (IQR), with whiskers extending to 1.5\*IQR.
* Outliers are shown as individual points.
* Use Case: Comparing exam scores across classes to identify outliers.
* Code Snippet:

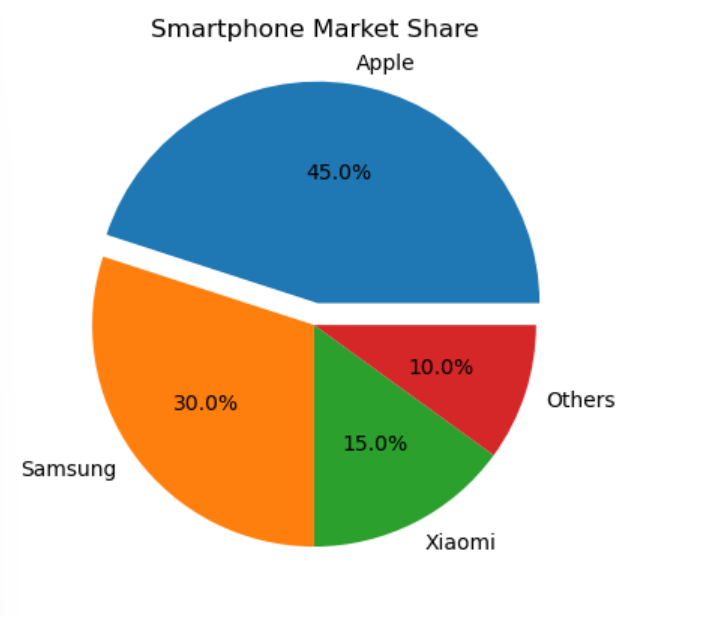


* Box Plot Output:
* Interpretation: A wider (class A) indicates greater score variability. The median line (inside the box) shows the central tendancy.

1. *Pie Charts:*

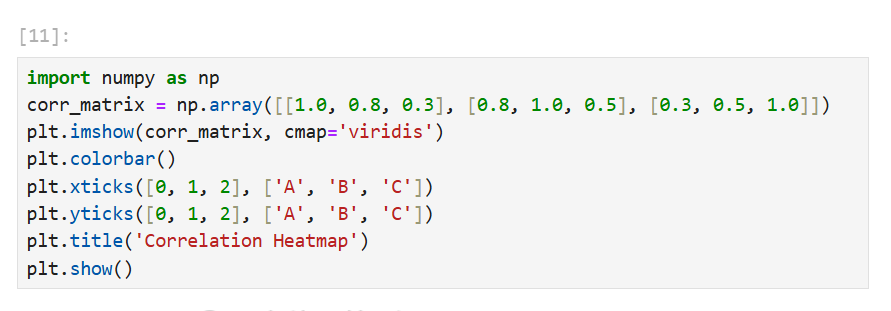
* Shows proportional data as slices of a circle.
* Use Case: Market Share Analysis.
* Code Snippet:



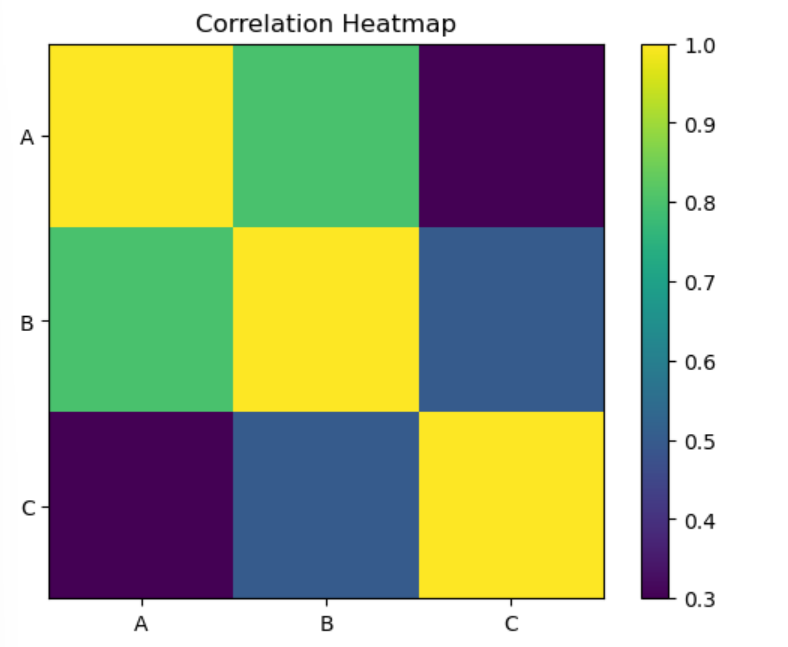
* Pie Chart Output:
* Interpretation: A circular chart with coloured slices; the “explode” parameter highlights the first slice.

1. *Heatmap:*

* Represents matrix data with colours.
* Use Case: Correlation matrix visualization.
* Code Snippet:



* Heatmap Output:



* Interpretation: A grid where colours (e.g., yellow for high values) indicate correlation strength.

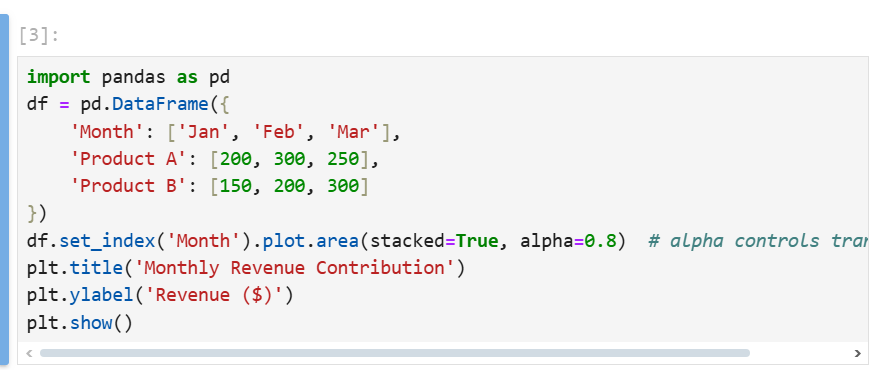
1. **Pandas:**

* Pandas, built on Matplotlib, simplifies data manipulation and rapid visualization directly from DataFrames.
* Its plot() method wraps Matplotlib functions, enabling quick insights during data exploration
* Unique Features:
* DataFrame Integration (Plot columns or rows without manual data extraction).
* Simplified Syntax (Generate plots with one-liners)
* Time-Series Support (Built-in handling of datatime indexes for time-based plots).

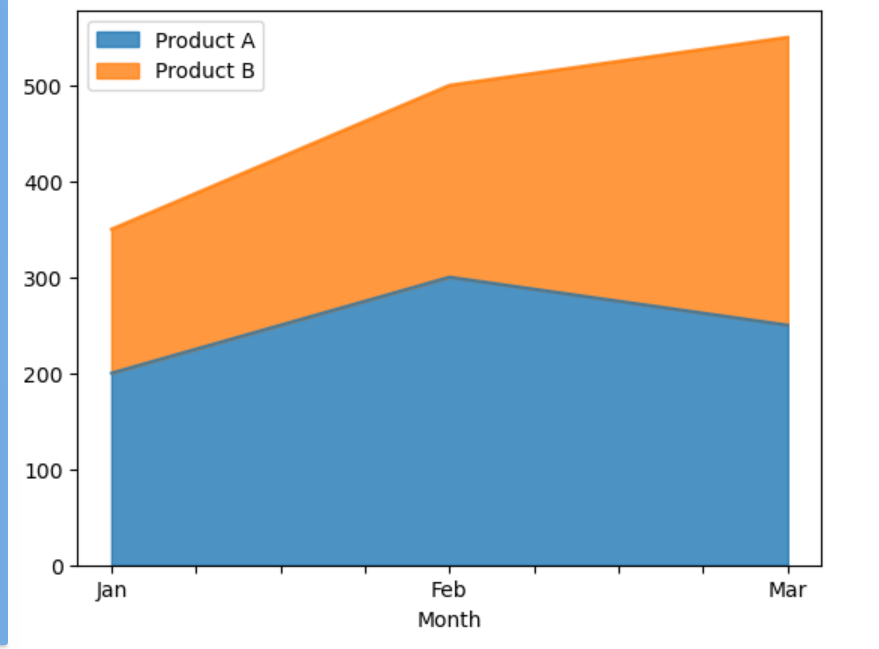
**Graph Types:**

1. *Area Plot:*

* A Stacked line plot where the area below the line is filled. Useful, for showing cumulative contributions of categories over time.
* Use Case: Visualizing monthly revenue contributions from multiple products.
* Code Snippets:



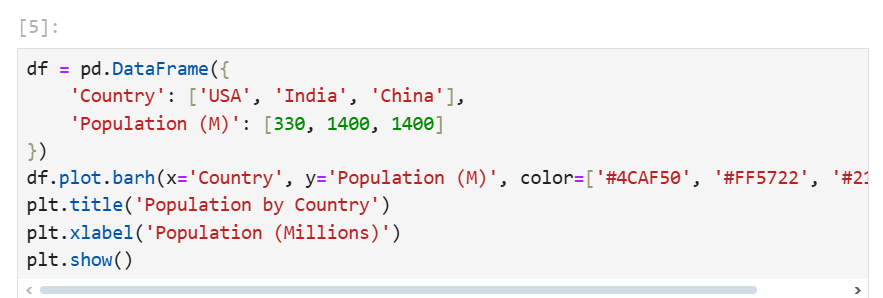
* Area Plot Output:



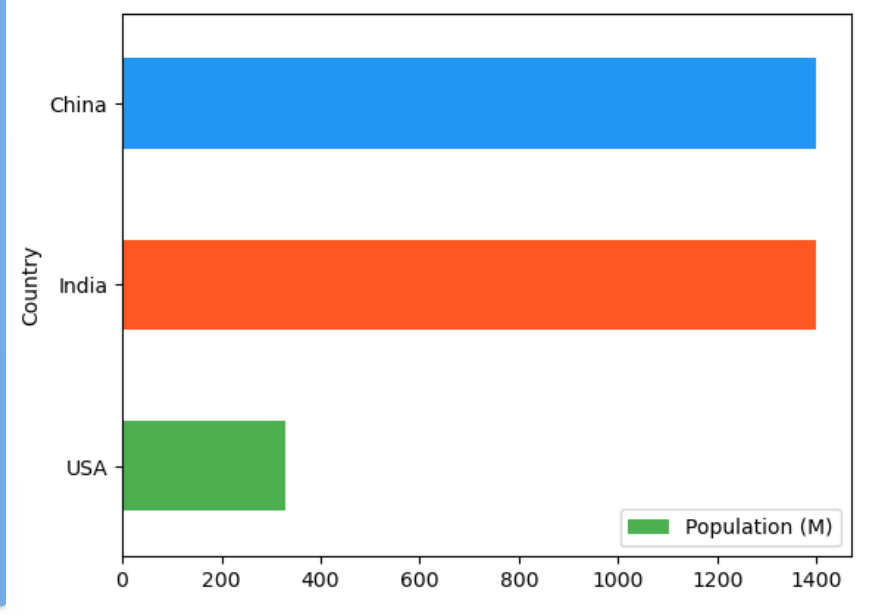
* Interpretation: The total height represents combined revenue, while coloured layers show individual product contributions

1. *Horizontal Bar Chart:*

* Horizontal bars compare values across categories.
* Useful when category names are long or when comparing many groups
* Use Case: Comparing population sizes countries.
* Code Snippet:



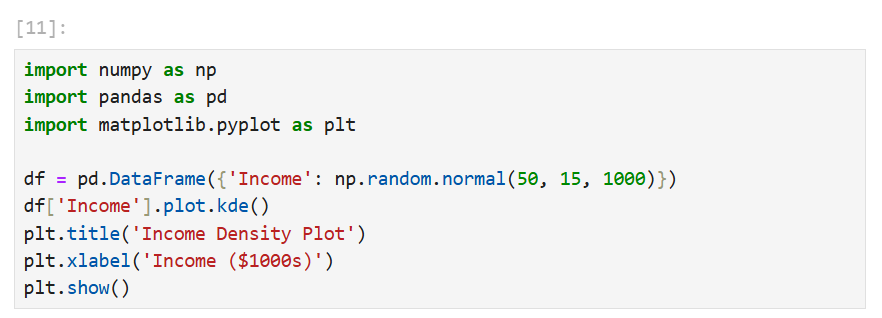
* Horizontal Bar Chart Output:



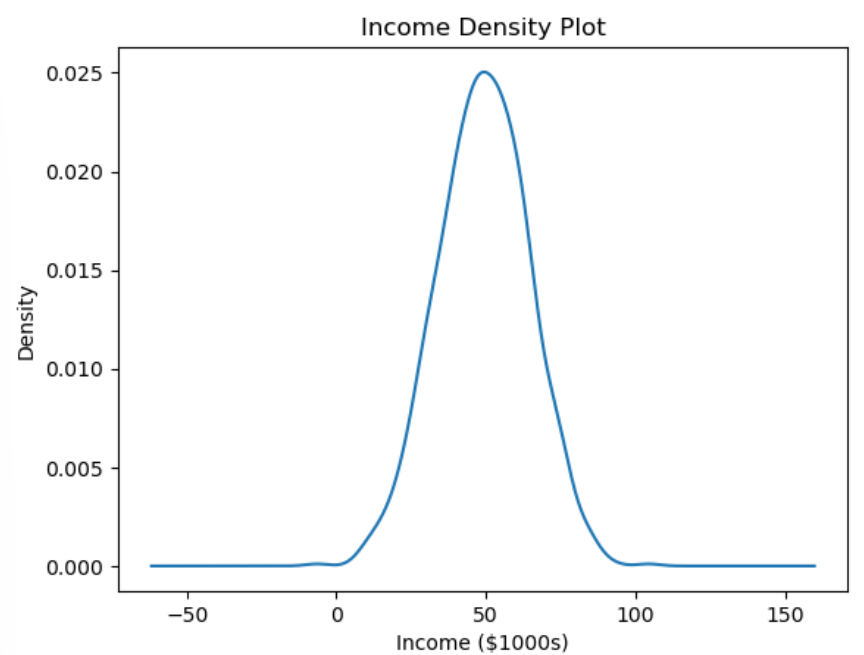
* Interpretation: Longer Bars (India and China) indicate larger populations. Colour differentiation aids quick comparison.

1. *Density Plot:*

* Smoothed histogram for data distribution
* Use Case: Analysing income distribution.
* Code Snippet:



* Density Plot Output:



1. **IN-DEPTH COMMPARISON**

|  |  |  |
| --- | --- | --- |
| Aspect | Matplotlib | Pandas |
| Ease of Use | Requires explicit code for customization | One-line syntax; minimal code for basic plots |
| Customization | Full control over aesthetics | Limited to DataFrame attributes |
| Interactivity | Supports interactivity with ‘mplcursors’ or integration with jupyter widgets | Static plots only; no hover or zoom tools. |
| Data Handling | Works with raw arrays/lists; no direct DataFrame integration. | Built for DataFrames; automatic alignment of indexed data. |
| Performance | Optimized for large datasets with efficient backend rendering. | Slower with massive datasets due to DataFrame overhead. |